# Digital Stock Market

## Binary Classification of Top 20 YouTube Videos

Jinsub Hong

June 24th, 2015

### Introduction:

The Digital Stock Market is a web application where users can buy virtual assets, or 'stocks' of Youtube videos, where the value is tied directly to the views the Youtube video has. Thus, one could buy and trade assets based on the viral potential of Youtube videos. This marketplace could be of use to advertisers so companies can gauge not only the viral potential of Youtube videos, but also gain a sense for how much advertising dollars should be spent on a video at a given point in time.

I seek to identify useful features in classifying YouTube videos that made it into the top videos of the week, in order to predict videos that will go viral in the future. This may either be used towards building an autonomous bidding agent for the Digital Stock Market, or simply identifying and purchasing advertising impressions for videos with viral potential.

I define a "viral video" as having made Youtube's top videos list (#popularOnYoutube), and non-viral as a random video pulled from Youtube that is not on that list. This was a simplifying assumption used such that we can discern what features the most viral videos exhibit.

| | | |
|---|---|---|
| Title | Title of the video | Unigram features |
| Description | Description of the video | Unigram features |
| Add count | Number of times this video was added | Integer |
| Likes | Number of likes | Integer |
| Dislikes | Number of dislikes | Integer |

| | | |
|---|---|---|
| Favorites | Number of favorites | Integer |
| Views | Number of views | Integer |

In this paper I suggest useful features currently active within the MYSQL database for the Digital Stock Market, combinations and alterations of these features, as well as identify outside features which were seemingly fruitful endeavors, but ended up contributing nothing.

## Data:

I pulled data for 60 videos, 30 of which represent a top video, and 30 of which represent a random video. These random videos were selected using randomyoutube.net, which provides a randomly generated link to a Youtube video. The top 30 videos were taken from the #popularOnYoutube channel, which represents the top trending videos on Youtube at the time. Both of these sets of videos were made at the beginning of the week, and updated every week to keep experimental results consistent, fresh, and not over-fit. The initial set of considered features were as follows:

## Model Descriptions:

I used Lightside to build three different types of classification models, naïve bayes, support vector machines, and logistic regression. Initially, they had classification rates of 61%, 64%, and 62% respectively. All experiments were conducted with 5-fold cross validation. They each had kappa statistics of 0.40, 0.41, and 0.40 respectively, suggesting that the balanced proportion of viral and non-viral videos yields classification that is not entirely by chance. In addition, I performed a standard student's t-test to test whether the performance difference between models statistically significant. These initial models only used the Title, Description, Likes, Dislikes, and Favorites count as features. While iterating through the models and their features, all three of SVM, naïve Bayes and Logistic regression were used.

SVM consistently came out on top with marginally better performance, but nevertheless statistically significant.

### Result of Error Analysis:

Descriptions and titles for many videos, particularly non-top videos, were often the same. Many top videos had similar unigram features between the title and the description (which makes sense, as the description frequently mentions the title). The differentiating factor between whether a video was top or not top, more often than not was not the unigrams present within, but rather the fact that the two fields were a copy of each other. For example, Jimmy Kimmel's video "Kid Explains Gay Marriage" (top video as of June 30th) has a very detailed description advertising not only his show, but also explaining what is going on in the video. Other videos, such as "Inside Out Trailer (Official)" although premium in content, were reposts, and did not have a descriptive description, and simply had "Inside Out Trailer (Official)" as the description. Another repost of the trailer simply had no description at all. Implementing these simple binary features yielded a model with a slight boost in performance (1%), but statistically different models under a t-test with a p-value of 0.02.

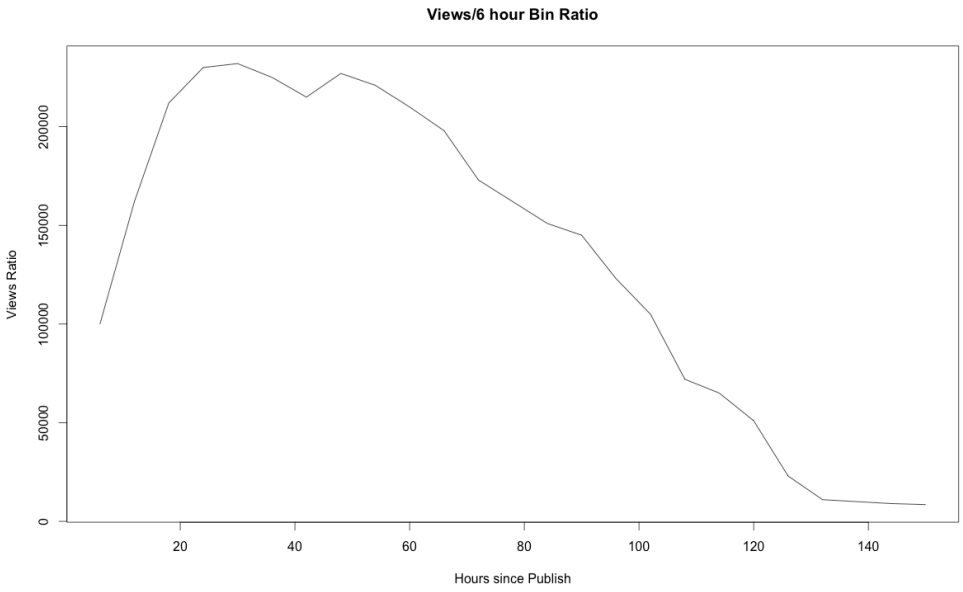
Divvying up the text fields via unigrams (splitting on white spaces) showed that numeric characters were highly weighted in determining a non-top video. This was because many videos that were picked at random had default video names, file extensions, or dates. However, not all numeric characters are part of a unigram and were thus interpreted as part of a normal string. Creating a separate binary feature checking for the presence of file extensions, and for numeric characters within the title yielded a performance increase of 1.5%, with statistically different models under a t-test with a p-value of 0.013.

"Views" was by far the most predictive feature, but is counter-productive to our goal, as we want to pinpoint constant features of videos that may be indicative of viral potential. So despite the predictive power of features such as likes, dislikes, comment, and favorites counts, they increase after a video starts to exhibit its viral potential. I thus deemed these features counterproductive and pulled them from the

models, as they cannot be taken into consideration without taking into account how long the video has been uploaded to Youtube. This underscored how important it was to take all features with respect to time, and not independently.

I attempted to create a feature to attempt to capture the velocity at which a video was obtaining views by diving the number of views by the number of days the video has been on Youtube. Binning by day revealed that the viral videos within a particular week have a ratio of at least 10000, and thus creating a binary feature with a cut off at 10000 was deemed a useful feature when added to the model under a t-test with a p-value of 0.01. However, it is worth noting that almost all top videos are added to Youtube within 7 days, meaning there is an extremely short window where this feature could be considered useful (perhaps only 1 or 2 days, as any longer the video will already be clearly viral and on the top Youtube videos channel). Focusing on videos that have only been in the top Youtube videos section for a maximum of 2 days reveals that they have a much higher views to days since publishing ratio, averaging 220000, as opposed to the approximate cut off of 100000, for all videos. This is a feature of how videos rapidly rise in popularity within the first few days, and taper off quickly by the end of the week. Perhaps a dynamic means of pulling videos, noting their views, and binning by hour would be a more useful endeavor, so as to capture videos right before they go viral. By tracking 10 example videos, we establish that this method may be useful in determining when a video is in the midst of its initial popularity spike.

Below is a plot approximating the views to day ratio of a randomly selected video from the top 100 on Youtube drawn on June 21$^{st}$, and tracked for 150 hours afterwards in 6-hour bins. In order to normalize the ratio to keep consistent with the views to day ratio mentioned earlier on, the ratios were multiplied by 4 to retain the same “per-day” scale. This was done for a total of 10 videos, and all exhibited much of the same behavior, peaking around 220000, and decaying rapidly after the second day. The below plot is representative of the 10 video sample size, though a larger, automatically generated sample would better reveal this type of behavior.




## Twitter Investigation:

Initially I suspected that using trending Twitter terms would yield possible bits of information regarding hot topics of conversation on the internet that would give clues as to what videos may go viral on Youtube. What I did not anticipate is how different the content on Twitter is from Youtube. It is without a doubt true that Twitter is in tune with hot topics of conversation in the news and in the public eye. Prominent issues such as the video of the police officer using excessive force to restrain an African American female during an unauthorized pool party in Mckinney Texas were all topics of conversation on Twitter, as well as links to copies of the video. However, despite how viral the video content was, and how prominent the issue was in the public eye, not a single copy of the video made its way into the top 100 videos on Youtube. In addition, after having manually tracked trending terms on Twitter for a month, there were only 11 overlapping terms. However, this was not a completely wasted effort as some terms were loosely related. For example, #ComicCon was a trending tag on Twitter. In addition, the trailer for the movie *Suicide Squad*, was within the top Youtube videos. This trailer was initially released at Comic Con. Perhaps a form of query expansion or a soft search would do well in connecting Twitter terms to videos. In the end, I argue that although there is an apparent lack of overlap in Twitter and Youtube's most viral terms and videos, there

may be ways to exploit soft-search methods to connect terms and videos, as there is occasionally some overlap in content (not explicitly in terms of trending tags and video titles) that could be exploited. For example, anticipated releases of products or media (such as the *Suicide Squad* movie) may be a category of Youtube videos or events that could be tracked.

### Further work and Overview:

Below is my final set of experimental features that were deemed useful for the model. The best performing model was an SVM, with 68% accuracy and a kappa statistic of 0.36.

| | | |
|---|---|---|
| Title | Title of the video | Unigram features |
| Description | Description of the video | Unigram features |
| Description length > Title length | Binary feature showing whether description is longer than the title | Binary |
| Description length < Title Length | Binary feature showing whether description is shorter than the title | Binary |
| Description length = title length | Binary feature showing whether description is the same length as title | Binary |
| Title contains numeric | Binary feature whether title has numeric character | Binary |
| Extensions | (.mp4, .mov, .avi, .mpeg4, .wmv, .flv) | Binary |
| Add count != 0 | Check to see if add-count was not zero | Binary |
| Views/days passed ratio > 10000 | #views / #days since publishing | Float |
| Top comment | The most up-voted comment | Unigram, Bigram features, stopword removal |

Incorporating more top comments and their respective text features may yield better results. Top videos have top comments that exhibit engaged, enthusiastic, excited language frequently beyond the single most up-voted

comment. Videos that do not do so well most often do not, and incorporating these features may help better differentiate top videos from non-top videos. Overall, most of the errors are being made where top-videos are deemed non-top, as expected. The best model had 58% of the errors being false-negatives. The ability to request the top comments, as of July 2015, is not an active Youtube API feature, though it is one of the most heavily requested ones.

A quick, manually added text field for the single most up-voted comment as a feature, using bigrams, unigrams, and stop word removal yielded an increased performance of 2% in a data set of size 60, with equal an equal number of viral and non-viral videos. A larger data set, and more than one comment would be a fruitful avenue to investigate when an automated way of adding top comments becomes available for each of the videos. As of now, the biggest barrier was manually finding the videos online and pasting the top comments into the CSV data set directly.

In the end, although attempts were made to tie top Youtube videos to current conversations in media by adding in trending Twitter terms, we find that the most fruitful features may be the ones that are closest to the videos themselves. In addition, these features need to be calculated and retrieved within the first week of the video being uploaded. This highlights the biggest challenge of this project: viral videos change constantly and videos have an extremely short life span. Thus, calculating any features must take this time window into account. In addition, comments may reveal more than any meta-features of the data, provided we can filter out the most important ones. Comments are where users most actively engage with a video and their reactions should play a more active role in how we predict viral potential.

Beyond simple unigram features on the comments, more semantic approaches to analyzing the video may be useful. Non-binary sentiment analysis could be one such approach where the amount of expressed excitement or positive engagement with a video is measured from the comments section. Another method could be to systematically identify which channels are prominent. It is easily observable that certain channel (such as the prominent *MovieClips Trailers* channel or *Jimmy Kimmel Live* channel) have videos with much more viral potential than

your average up-loader. Finally, certain categories or topics of videos have much more viral potential than others, for example, TV skits, or movie trailers, or music videos. If a simple topic model could be used to classify or create likelihoods for videos' categories, induced from the comments and descriptions, one could perhaps better predict viral videos.

## Reflection:

When I first approached this project, I was in a rush for answers. Spurred by a desire for immediate results, I set about experimenting with features to boost cross-validated performance. I fixated on kappa statistics and accuracy measures, and endlessly searched through Youtube for possible feature candidates to address my error analysis. I stuck with a static data set and neglected the temporal nature of all of the features and ended up with radically high numbers. I had over-fit not only to the static data set, but also to features that were useful because the video was already viral (such as views, and favorites). I wish I had asked myself earlier what role time played in engineering my features rather than blindly searching for useful ones. This way I would have realized earlier on that almost all numerical features were relative to time, rather than independent.

I also believe I was fixated on numerical features, and overlooked the potential inherent in the comments. Had I had more time or a team, I believe the problem of filtering out top comments (although not provided by the Youtube API) could have been solved internally. This would have allowed for more creative features and a lot more text analysis.

My biggest pitfall however was my lack of documentation regarding my process. Again, I was so fixated on results, I gave little thought to reproducibility. Had I thought of that before embarking on my project, I would have created backups, and created more scripts (instead of cranking out hard work via Excel).

Overall this project was an invaluable learning experience, the independence and the accompanying potential for failure made me a much smarter scientist, as opposed to merely a harder working one.

## Appendix:

With the exception of the views/days ratio feature, and the description length binary features, all features were created in LightSide either using the Unigram & Bigram boxes, or the regular expression function. The regular expression function was used to create binary features containing specific strings such as ".mp4" or the numeric characters in text fields such as the description or title. This was tedious, yet was a logical progression in my search to experiment and see which phrases were useful or not. The views/day ratio feature was calculated using simple manipulations in Excel that could easily be replicated. Finally, the description length feature was created using a simple Python script which simple printed out whether the title string was shorter or longer than the description. Below is a screenshot of one of the models that were tested and trained on different weeks. This was an SVM, and was the best performance out of all of my models.

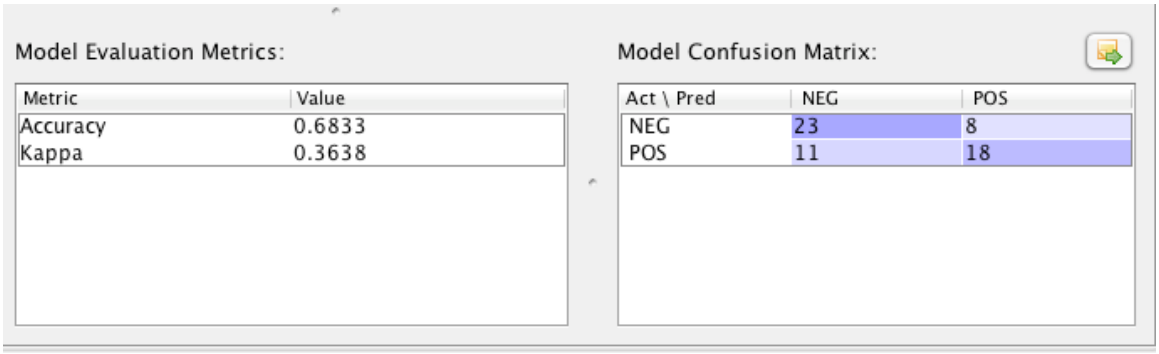



**Figure 1: Model Trained On June 21st, Tested on July 10th**

Below is a description of my available artifacts:

*joinPull.bash* – A simple join statement joining two tables in the database to have all needed information in one place. Processing needed to create features from this set and possible de-duplication

*6302015.csv* – An example dataset pulled by the above command.

*titleFeatMaker.py* – A script that created the title length feature